CLAIMS

1. A software architecture for a distributed computing system comprising:

an application configured to handle requests submitted by applications executing on remote devices over a network; and

an application program interface to present functions used by the applications to access network and computing resources of the distributed computing system, wherein the application program interface comprises a set of base classes and types that are used in substantially all applications executing on the remote devices submitting requests.

2. A software architecture as recited in claim 1, wherein the set of base types comprises:

an AsyncCallback delegate supplied to an application, wherein the AsyncCallback delegate references a callback method to be called when a corresponding asynchronous operation is completed; and

an IAsyncResult interface that enables determination of the status of an asynchronous operation.

3. A software architecture as recited in claim 2, wherein the IAsyncResult interface includes:

an AsyncState property that returns the object that was provided as the last parameter as part of a Begin call corresponding to the asynchronous operation;

an AsyncWaitHandle property that returns a WaitHandle that can be used to allow the application to wait for a call to be completed without needing to poll;

a CompletedSynchronously property that is set to true if the Begin call corresponding to the asynchronous operation completed synchronously; and

an IsCompleted property that is set to true after processing of the asynchronous operation is completed.

4. A software architecture as recited in claim 1, wherein the set of types support an event model including an event delegate that connects an event with a handler of the event, the set of base classes and types further comprising:

one or more classes that hold event data; and

one or more delegates that identify a method to provide a response to an event.

5. A software architecture as recited in claim 1, wherein the application program interface further comprises a collections namespace that includes a plurality of classes and interfaces for in-memory data storage and manipulation.

6. A software architecture as recited in claim 5, wherein the collections namespace includes, as at least part of the plurality of types:

a first set of types including commonly used collection classes;

a second set of types including interfaces to define a formal contract between developers creating new collections and developers consuming collections; and

a third set of types that support creating strongly typed collections.

7. A software architecture as recited in claim 1, wherein the application program interface further comprises a globalization namespace that includes a plurality of classes that define culture-related information, wherein the plurality of classes include a first set of types representing information about a user's culture and a second set of types representing information about a user's region.

8. A software architecture as recited in claim 1, wherein the application program interface further comprises a net namespace that includes a plurality of classes that enables use of network resources without details of one or more protocols used to access the network resources.

9. A software architecture as recited in claim 1, wherein the application program interface further comprises a security namespace that includes a plurality of classes and interfaces that make available an underlying structure of a security system including one or more cryptographic services, code access security and role based security infrastructure.

10. A software architecture as recited in claim 1, wherein the application program interface further comprises a service process namespace that includes a plurality of classes that allow installation and running of services.

11. A software architecture as recited in claim 1, wherein the application program interface further comprises a serialization namespace that includes a plurality of classes that enable serializing and deserializing of instance data.

12. A software architecture as recited in claim 1, wherein the application program interface further comprises a diagnostics namespace that includes a plurality of classes that enable debugging of applications, trace code execution, reading event logs, writing event logs, and monitoring system performance.

13. A software architecture as recited in claim 1, wherein the application program interface further comprises a messaging namespace that includes a plurality of classes that enable connecting to message queues on the network, sending messages to message queues, receiving messages from message queues, and peeking at messages from message queues.

14. An application program interface, embodied on one or more computer readable media, comprising:

an AsyncCallback delegate supplied to an application, wherein the AsyncCallback delegate references a callback method to be called when a corresponding asynchronous operation is completed; and

an IAsyncResult interface that enables determination of the status of an asynchronous operation.

15. An application program interface as recited in claim 14, wherein the IAsyncResult interface includes:

an AsyncState property that returns the object that was provided as the last parameter as part of a Begin call corresponding to the asynchronous operation;

an AsyncWaitHandle property that returns a WaitHandle that can be used to allow the application to wait for a call to be completed without needing to poll;

a CompletedSynchronously property that is set to true if the Begin call corresponding to the asynchronous operation completed synchronously; and

an IsCompleted property that is set to true after processing of the asynchronous operation is completed.

16. A distributed computer software architecture, comprising:

one or more applications configured to be executed on one or more computing devices, the applications handling requests submitted from remote computing devices;

a networking platform to support the one or more applications; and

an application programming interface to interface the one or more applications with the networking platform, wherein the application program interface comprises a set of types that are used in each of the one or more applications.

17. A distributed computer software architecture as recited in claim 16, wherein the set of base classes and types comprises:

an AsyncCallback delegate supplied to an application, wherein the AsyncCallback delegate references a callback method to be called when a corresponding asynchronous operation is completed; and

an IAsyncResult interface that enables determination of the status of an asynchronous operation.

18. A method comprising:

receiving one or more application program interface (API) calls from one or more remote devices over a network, wherein the one or more calls are to one or more functions that include a set of base classes and types that are used in substantially all applications executing on the one or more remote devices; and

performing the function requested in each of the one or more calls.

19. A method as recited in claim 18, wherein the set of base classes and types comprises:

an AsyncCallback delegate supplied to an application, wherein the AsyncCallback delegate references a callback method to be called when a corresponding asynchronous operation is completed; and

an IAsyncResult interface that enables determination of the status of an asynchronous operation.

20. A method as recited in claim 18, wherein the set of base classes and types support an event model including an event delegate that connects an event with a handler of the event, the set of base classes and types further comprising:

one or more classes that hold event data; and

one or more delegates that identify a method to provide a response to an event.

21. A method as recited in claim 18, wherein the application program interface further comprises a collections namespace that includes a plurality of classes and interfaces for in-memory data storage and manipulation.

22. A method as recited in claim 18, wherein the application program interface further comprises a globalization namespace that includes a plurality of classes that define culture-related information, wherein the plurality of classes include a first set of types representing information about a user's culture and a second set of types representing information about a user's region.

23. A method as recited in claim 18, wherein the application program interface further comprises a net namespace that includes a plurality of classes that enables use of Internet resources without details of one or more protocols used to access the Internet resources.

24. A method comprising:

calling, to one or more remote devices over a network, one or more functions via an application program interface (API) that make available a set of

base classes and types that are used in substantially all applications calling the one or more functions;

receiving, from the one or more remote devices, a response to the calling.

25. A method as recited in claim 24, wherein the set of base classes and types comprises:

an AsyncCallback delegate supplied to an application, wherein the AsyncCallback delegate references a callback method to be called when a corresponding asynchronous operation is completed; and

an IAsyncResult interface that enables determination of the status of an asynchronous operation.

26. A method as recited in claim 24, wherein the set of types support an event model including an event delegate that connects an event with a handler of the event, the set of base classes and types further comprising:

one or more classes that hold event data; and

one or more delegates that identify a method to provide a response to an event.

27. A method as recited in claim 24, wherein the application program interface further comprises a collections namespace that includes a plurality of classes and interfaces for in-memory data storage and manipulation.

28. A method as recited in claim 24, wherein the application program interface further comprises a globalization namespace that includes a plurality of classes that define culture-related information, wherein the plurality of classes include a first set of types representing information about a user's culture and a second set of types representing information about a user's region.

29. A method as recited in claim 24, wherein the application program interface further comprises a net namespace that includes a plurality of classes that enables use of Internet resources without details of one or more protocols used to access the Internet resources.